WEST

Generate Collection Print

L2: Entry 22 of 27 File: USPT Apr 18, 2000

DOCUMENT-IDENTIFIER: US 6052681 A
TITLE: X.500 system and methods

Detailed Description Text (192):
A filter is a combination of one or more filter items connected by the operators AND, OR and NOT. For example; surname="MASTERS" AND title="SALES MANAGER"

Current US Original Classification (1):
707/3

Current US Cross Reference Classification (1):
707/4

WEST

Generate Collection Print

L2: Entry 25 of 27 File: USPT Jun 25, 1996

DOCUMENT-IDENTIFIER: US 5530853 A
TITLE: Method for filtering items in a computer application program container object using filter data for related entries in a container object of another application program

Brief Summary Text (2):
The present invention relates to filtering items in a computer application program container object. By filtering is meant comparing data contained one or more specified item data fields with user-entered filter data, and selecting only those items whose data match the filter data. More specifically, the present invention relates to an improved filtering method which allows a computer user to filter such items using data contained in the items and data contained in an address book data base.

Brief Summary Text (3):
Computer users frequently find it advantageous to filter the items in a computer application program container object, such as a file cabinet, a file cabinet drawer, a folder within a file cabinet drawer, or an electronic mail in-basket or out-basket. For example, a user might desire a list of all the items contained by a file cabinet which were sent by a specific person, i.e., items in which that specific person is the addressor. Such a list may be compiled by using the person's name to filter the addressor data fields of each item.

Brief Summary Text (4):
Presently, a computer user may filter items using filter data for data contained in the items. Thus, a computer user may typically filter the items using filter data for such data as the addressor's name, an addressee's name, the subject of the item, the date of the item, and the name of a person to whom a copy of the item was sent.

Brief Summary Text (5):
The present invention allows a computer user to filter items in a container object using filter data for data contained in an address book data base and data contained in the items. As address book data bases may contain such additional data as a person's department, address, and status (management, nonmanagement, etc.), the invention provides an extended filtering method which is much more flexible and comprehensive than methods presently available.

Brief Summary Text (7):
An extended method for filtering items contained in a computer application program container object is provided which allows the computer's user to filter the items using data contained in the items and data contained in an address book data base.

Brief Summary Text (9):
The items in the selected container object are then filtered with the item-based filter data, and the items which pass that filter are listed on a temporary list. If no item-based filter data is entered by the user, all the items in the container are listed on the temporary list.

Drawing Description Text (4):
FIG. 2 is a pictorial representation illustrating the format of an address book data base entry which may be used to filter the item of FIG. 1;

Detailed Description Text (10):
In the example, the user entered filter data ("Expense Reports") in an item-based filter data field (Subject 35). Thus, in the example, the program proceeds to block 109. If the user had not entered item-based filter data at block 103, the program would continue to block 107, where the program would list the items in the file cabinet on a

temporary list.

Detailed Description Text (11):
At blocks 109 and 111, the program filters the items in the file cabinet with the item-based filter data and lists those items which pass the filter on the temporary list. Techniques for filtering items are well known in the art and will not be discussed here.

Current US Original Classification (1):
707/1

WEST

Generate Collection Print

L2: Entry 26 of 27 File: USPT May 31, 1994

DOCUMENT-IDENTIFIER: US 5317742 A
TITLE: Dynamic translation of network management primitives to queries to a database

Drawing Description Text (9):
FIG. 8 is a flow chart of the processing of a filter item in the present invention.

Detailed Description Text (105):
The remaining processing depends on the outcome of this examination at 374. If the choice tag is a filter item, the filter item is processed at 380 and control returns to step 364. If the choice tag is AND, OR, or NOT, control passes to 382, 384 or 386 respectively where the appropriate placeholder clause is generated. Control passes from 382, 384 or 386 to step 390 where the current CMISFilter context is saved on the stack and processing begins on the next nested CMISFilter. Control then returns to step 364.

Detailed Description Text (118):
The individual processing of each filter item is described in detail in conjunction with FIG. 8 beginning at step 400. Then, the routine determines if the specified attribute ID in the filter item is multi-valued at step 402. If so, the routine processes the set-valued attribute at 404 (see FIG. 9) and exits at 408. If not, control passes to step 412 where the table and column corresponding to the particular attribute is identified. To process each individual FilterItem in the CMISFilter, the choice tag of the FilterItem is examined at step 416. If a FilterItem is a comparison to be performed with respect to a specific attribute at 416, the routine generates a SQL clause for comparison of the attribute at step 420 (see FIG. 10) and then exits at 408. If a FilterItem is an existence operation at 416, the routine generates a SQL "column-name IS NOT NULL", where column-name is the name of the column in the table corresponding to the specified attribute, and places it in the allocated buffer at 422 and then exits at 408. If a FilterItem is a substring operation at 416, the appropriate SQL clause for substring operation is generated at 426 (see FIG. 11) and the routine exits at 408.

Detailed Description Text (119):
Turning to FIG. 9, the set-valued attribute process of step 404 is described in greater detail starting at 440. The routine determines the dependent subtable in which the set-valued attribute is stored at step 442, then examines the choice tag on the filter item at 446. Process control then goes to step 450, 452 or 454 for generation of the SQL clause depending upon whether the asserted value of the choice tag is a subset of the actual value, a non-null intersection with the actual value, or a superset of the actual value respectively. The SQL clause is then placed in the buffer at 458 and the routine exits at 460.

Detailed Description Text (130):
If the choice tag is 1, a substring comparison is to be performed between the asserted value and the actual attribute value stored in the MIB. The nature of the substring comparison depends on the tag of the nested CHOICE type in the filter item:

Current US Original Classification (1):
707/3

Current US Cross Reference Classification (1):
707/4

the choice of an execution plan significantly. Finally, if the conditions in the filter condition are not independent, the problem of determining an optimal search-minimal execution is NP-hard.

Detailed Description Text (11):
The present invention solves the problem of optimizing the evaluation of a ranking expression. Previous significant work in this area is due to Fagin's work. A key contribution of the present invention is to show that ranking expressions can be processed "almost" like filter conditions without any expected loss of efficiency. Indeed, it is shown here that the technique of the present invention is expected to retrieve no more objects than the strategy developed by Fagin. In fact, experimental results as explained in Section 5 below show that the performance gain from processing ranking expressions as filter conditions according to the present invention can reach over 30% in certain circumstances.

Detailed Description Text (13):
The rest of the specification is organized as follows. Section 2 describes the query model utilized in a preferred embodiment of the invention. Sections 3 and 4 present the results of evaluating filter conditions and ranking expressions, respectively. Finally, Section 5 discusses the performance gain realized using the prototype implementation of the invention.

Detailed Description Text (19):
The second requirement of the query model constructed according to the present invention is fulfilled by allowing a user to designate a "filter condition". A filter condition is a set of requirements defined during the search which an object must fulfill in order to be selected. For instance, a filter condition to find a picture having a field of yellow and red flowers may have the filter conditions specified as color-yellow.gtoreq.0.3, color-red.gtoreq.0.4, and text-"flower".gtoreq.1.0 where the text is an exact match for the filter condition. This filter condition has three subconditions or atomic filter conditions. An object satisfies these subconditions if the grade of match for the particular attributes is at least the grade values specified (e.g., color-yellow.gtoreq.0.3, color-red.gtoreq.0.4, and text-"flower".gtoreq.1.0). Additional filter conditions are generated from the atomic conditions by using the AND and OR boolean connectives (evaluated as either true or false) in the method described further below.

Detailed Description Text (22):
The above query asks for k objects in the object Repository with the highest grade for the Ranking.sub.-- expression, among those objects that satisfy the Filter condition. Intuitively, the filter condition eliminates unacceptable matches, while the ranking expression orders the acceptable objects.

Detailed Description Text (37):
Other popular attributes are features of images. If an object of a repository contains an image, an attribute of the object could be the color histogram of that image. Then, a filter condition query on such an attribute can ask for objects whose image histogram matches a given color histogram closely, for example. One conventional method of handling such attributes and queries is by using R trees and its variants to index the feature vectors associated with the attributes. The grade between two feature vectors is computed based on the semantics of the attributes using sophisticated algorithms which are already known in the art.

Detailed Description Text (39):
3. Filter Conditions

Detailed Description Text (40):
The following section describes the application of the methods of the present invention to the processing and cost-based optimization of queries having only a filter condition, i.e., the queries are of the form:

Detailed Description Text (44):
For illustrative purposes, assume that the repository requires the use of an index to evaluate every atomic filter condition. One way to process such queries is to retrieve object ids using one GradeSearch for each atomic condition in the filter condition, and then merge these sets of object ids through a sequence of unions and intersections. An alternative is to retrieve a set of object ids using GradeSearch for some conditions, and check the remaining conditions on these objects through probe operations.

Detailed Description Text (45):
The key optimization problem is to determine the set of filter conditions that are to be evaluated using GradeSearch. The rest of the conditions will be evaluated by using Probe. In order to efficiently execute the latter step, the present invention exploits the known techniques in optimizing the probing of expensive filter conditions as in Predicate Migration: Optimizing Queries with Expensive Predicates, Proceedings of the ACM SIGMOD International Symposium on Management of Data, Washington D.C. (May 1993).

Detailed Description Text (46):
The first step of optimizing queries having only filter conditions is to define a space of search-minimal executions, and sketch the cost model and the optimization criteria. The next step is to construct an optimization algorithm from these criteria taking into account the conditions for optimality. Finally, the execution plan produced by the algorithm can be improved through post-optimization. Considering both the search and probe costs leads to significantly better execution strategies, where post-optimized search-minimal executions behave almost as well as the best (not necessarily search-minimal) executions. However, it has been found that the general problem of determining an optimal search-minimal execution is NP-hard, i.e., that it is extremely unlikely that one could build efficient solutions for this general problem.

Detailed Description Text (48):
The following is a discussion of the possible space of execution for simple filter conditions, i.e., conditions that consist of a disjunction (OR) or a conjunction (AND) of atomic conditions. The possible space of execution can be generalized as described below for arbitrary filter conditions.

Detailed Description Text (50):
Consider now the case where the filter condition is a disjunction of atomic filter conditions 1 through n: a.sub.1 OR . . . OR a.sub.n, where a.sub.i is used as a shorthand for an atomic condition specifying an attribute, value, and grade. All objects that satisfy at least one of the subconditions satisfy the entire filter condition. Evaluation of an atomic condition a.sub.i requires the use of the GradeSearch access method associated with a.sub.i. Thus, if the atomic conditions are independent (defined more precisely in Section 3.2 below), then use of a GradeSearch is needed for each atomic condition so as not to miss any object that satisfies the entire condition.

Detailed Description Text (51):
Consider now the case where the filter condition is a conjunction of atomic filter conditions a.sub.1 AND . . . AND a.sub.n. There are several execution alternatives. In particular, all the objects that satisfy the filter condition can be retrieved by using GradeSearch on any one of the atomic conditions a.sub.1, . . . a.sub.n. Subsequently, each retrieved object can be tested to verify that it satisfies all of the remaining subconditions. The cost of using one atomic condition for GradeSearch instead of another may lead to significant differences in the cost. Thus, the conjunction of atomic filter conditions can be processed by executing the following steps:

Detailed Description Text (55):
The above class of execution alternatives for a conjunctive query is called "search-minimal" since only a minimal set of subconditions (in this case, only one subcondition) is used for GradeSearch. The search-minimal strategies represent a subset of the possible executions. In particular for a conjunctive filter condition, instead of searching on a single subcondition and probing on the others, it is possible to search on any subset of the atomic conditions and to take the intersection of the object-ids retrieved. However, the space of all such executions is significantly larger. In particular, there are exponentially many subsets of conjuncts to search on, but only a linear number of minimal conjunct sets for searching.

Detailed Description Text (56):
A search-minimal execution performed according to the present invention evaluates a minimal set of atomic conditions using GradeSearch, and evaluates the rest of the conditions using Probe. A simple conjunctive filter condition requires a GradeSearch over only one atomic condition. However, an arbitrary filter condition involving AND's and OR's might need to search more than one atomic conditions, like the disjunction above.

Detailed Description Text (58):
A set of objects are obtained when searching on a subcondition using GradeSearch. In certain situations, the invention does additional probing to determine the subset of

objects that satisfy the entire filter condition. Thus, given an atomic condition and a filter condition, the "residue" is the condition that the objects retrieved through the subcondition should pass to satisfy the entire filter condition. The following definition shows how residues are constructed using a tree representation of the filter conditions.

Detailed Description Text (60):
Let f be a filter condition, represented as a tree, and a an atomic condition of f. Consider the path from the leaf node for a to the root of the tree for f. For every .andgate. node i in this path, let a.sub.i be the condition consisting of the conjunction of all the subtrees that are children of the node i and that do not contain a. Then the residue of f for a, R(a,f), is .andgate..sub.i a.sub.i.

Detailed Description Text (62):
Consider the filter condition:

Detailed Description Text (63):
Using the above definition of residue as applied to the atomic condition a.sub.2, a.sub.1 =a, and a.sub.2 =a.sub.4. Hence, R(a.sub.2,f)=a.sub.1 .andgate.a.sub.4. If a.sub.4 is chosen as the search condition, then R(a.sub.4,f)=((a.sub.1 .andgate.a.sub.2).orgate.a.sub.3). Then, any object that satisfies a.sub.4 and also satisfies R(a.sub.4,f) satisfies the entire filter condition.

Detailed Description Text (64):
Given a filter condition f the optimization method practiced according to the present invention characterizes the smallest sets of atomic conditions such that by searching these conditions, all of the objects that satisfy f are retrieved (plus some extra ones that are pruned out by probing).

Detailed Description Text (66):
A complete set of atomic conditions m.sub.f for a filter condition f is a subset of the atomic conditions in f such that any object that satisfies f also satisfies at least one of the atomic conditions in m.sub.f. A complete set m.sub.f for f is a search-minimal condition set if there is no proper subset m.sub.f ' of m.sub.f that is also complete.

Detailed Description Text (74):
A search-minimal execution of a filter condition performed according to the present invention searches the repository using a search-minimal condition set A, and executes the following steps:

Detailed Description Text (89):
Consider first a textual attribute that is handled by a vector-space retrieval system. Typically such a system has inverted lists associated with each term in the vocabulary. Two numbers can be extracted from each list: the number of documents that contain the corresponding term, and the added weight of the term in the documents that contain it. Using these extracted numbers, the selectivity of an atomic filter condition, as well as the cost of processing the inverted lists can be easily estimated.

Detailed Description Text (92):
The optimization for only a particular class of filter conditions is described below, although other class optimizations are possible using the teachings of the invention. Similar restrictions have been traditionally adopted since the System-R optimization effort.

Detailed Description Text (94):
A filter condition f is "independent" if:

Detailed Description Text (95):
1. Every atomic filter condition occurs at most once in f

Detailed Description Text (96):
2. Every n atomic filter conditions e.sub.1, . . . , e.sub.n satisfy the following constraint: ##EQU2## where p(e) is the probability that the filter condition e is true.

Detailed Description Text (97):
When the filter condition is independent, the selectivities of complex filter conditions can be computed using the following two rules as in traditional

optimization:

Detailed Description Text (101):
The present invention optimizes filter conditions according to the following steps which are further described below. First, an optimization metric is defined over the search-minimal execution space. Next, a strategy for probing is determined using the past work in optimizing boolean expressions. Though the invention is optimal for independent filter conditions, it can be adapted for non-independent filter conditions using the methods described below. It is shown that if the filter condition is not independent, then the complexity of determining an optimal execution is NP-hard.

Detailed Description Text (103):
The following definition for the cost of search-minimal executions enables a user to pick the least expensive search-minimal execution. The cost of one such execution depends on (a) the choice of the search conditions, (b) the probing costs of the remaining conditions, and (c) the cost of taking the union of the answer sets. The cost of (c) dominates only when the selectivity of the filter condition is low. Therefore, to simplify the optimization problem, the method for cost-based optimizing practiced according to the invention focuses only on the search and probe costs.

Detailed Description Text (104):
Given a search-minimal condition set m for a filter condition f, the cost of searching the conditions in m plus the cost of probing the other conditions can be computed, as follows: ##EQU3## where .vertline.O.sub.a .vertline. is the number of objects that satisfy condition a. Thus, if there are O objects in the repository, .vertline.O.sub.a .vertline.=Sel(a)*O. Note that crucial to the above cost estimation is the ability to compute the probing cost PC(R(a, f),.vertline.O.sub.a .vertline.)). The estimation of such a cost depends on the strategy used to evaluate R(a, f).

Detailed Description Text (109):
Consider the filter condition a.sub.1 .andgate.a.sub.2 .andgate.a.sub.3, where Sel(a.sub.1)=0.01, Sel(a.sub.2)=0.02, and Sel(a.sub.3)=0.05. Let c.sub.1 =c.sub.2 =1, and c.sub.3 =0.5. The increasing rank sequence is then c.sub.3,c.sub.1,c.sub.2. Then, the probing cost for 1000 objects is as follows:

Detailed Description Text (114):
Let M be the set of all search-minimal condition sets for a filter condition f. A search-minimal condition set m is optimal if and only if C(f, m)=min.sub.i.epsilon.M C(f, i).

Detailed Description Text (115):
The preferred method for determining the optimal search-minimal condition set for an independent filter condition is described below. The algorithm is implicit in the following inductive definition, where the algorithm traverses the condition tree in a bottom-up fashion to create the optimal set of search-minimal conditions.

Detailed Description Text (117):
Let f be a filter condition. The "inductive" search-minimal condition set for f, bm(f), is defined inductively for a subcondition f' as follows:

Detailed Description Text (122):
Let f be an independent filter condition. Then bm(f) is an optimal search-minimal condition set.

Detailed Description Text (124):
The filter expression f can be viewed as an expression tree since f is independent, and thus contains no repeated atomic conditions. The proof is by induction on the height of the tree, and uses the fact that since f is independent, if f.sub.1 and f.sub.2 are children of an OR or an AND node, the optimal search-minimal condition set may be created from the search-minimal condition sets for f.sub.1 and f.sub.2.

Detailed Description Text (125):
The proof of optimality of bm(f) depends on the fact that the given filter condition is independent. Nonetheless, with the following simple modification, the above method still provides a search-minimal condition set in case the given condition is not independent. However, this set is no longer guaranteed to be optimal:

Detailed Description Text (130):
Let us assume that the filter condition is (a AND b) OR (a AND c). The first step of

the algorithm treats every instance of a as a different condition. So, the query is viewed by Step (1) as (a.sub.1 AND b) OR (a.sub.2 AND c). Assume that the method determines bm(f)={b,c}. Step (2) of the method does not change bm(f), although {a} could be a significantly better search-minimal condition set. Therefore, the above method may fail to identify the best search-minimal condition set if the subconditions are not independent, as in this example.

Detailed Description Text (133):
The problem of determining an optimal search-minimal condition set for a filter condition is NP-hard.

Detailed Description Text (135):
The result can be proved by a reduction from the vertex-cover problem. To map an instance of the vertex-cover problem G to our problem, a filter condition F is generated such that G has a vertex cover of size k or less if and only if there is a processing strategy for F that retrieves objects using searches over k or fewer atomic conditions. A unit cost is associated for every search, and zero cost for the probes to complete the proof.

Detailed Description Text (138):
First, a search-minimal execution for a filter condition according to the invention always handles the residue of a search subcondition by probing. However, when the number of objects to be probed is high, the cost of probing may exceed the cost of searching on the atomic condition using GradeSearch. Thus, in case of a conjunctive query, it might be more efficient by selecting more than one condition for searching. In other words, it could be convenient to allow the conditions that are used for searching to no longer form a search-minimal condition set. However, the present optimization method does not consider such a plan. In one embodiment of the invention, a post-optimization step locally replaces probes on one or more conditions by the corresponding searches, as described by the strategy below.

Detailed Description Text (142):
4. Filter Conditions and Ranking Expressions

Detailed Description Text (143):
This section describes the application of the present invention to queries that consist not only of a filter condition, but also of a ranking expression. The answer to such queries consists of the top objects for the ranking expression that also satisfy the filter condition. The method for constructing queries consisting just of ranking expressions is described in Section 4.1. Section 4.2 describes a prior art method for processing these types of queries.

Detailed Description Text (145):
As described in further detail below, the ranking expression can be mapped into a filter condition and processed approximately as if were a filter condition, with no expected performance loss. Using this result, queries with ranking expressions can be processed applying the methodology described above in Section 3 for processing filter conditions. The experimental results of Section 5 support this result by showing that the number of objects retrieved when processing a ranking expression as a filter condition can be considerably smaller than when processing the ranking expression using the methods found in the prior art.

Detailed Description Text (157):
4.3 Processing Ranking Expressions as Filter Conditions

Detailed Description Text (158):
Ranking expressions can be processed as modified filter conditions by considering the space of search-minimal executions for filter conditions, without any expected loss of efficiency, thus providing an elegant and uniform treatment of ranking expressions and filter conditions.

Detailed Description Text (159):
In the novel method described below, a ranking expression is specified to get a sorted set of objects. Filter conditions have flat sets as their answers. In the strategy described below, after processing a ranking expression as a filter condition, the grade of the retrieved objects for the ranking expression is computed and the retrieved objects sorted before returning them as the answer to the query.

Detailed Description Text (161):

1. There exists a method to assign a grade for each atomic expression in the ranking expression, and a filter condition with the same "structure" as the ranking expression such that the filter condition retrieves at least a desired number of top objects.

Detailed Description Text (162):
2. There is a search-minimal execution for the filter condition that retrieves a sufficient number of objects that is no larger than the expected number of objects that Fagin's method would retrieve.

Detailed Description Text (166):
In other words, the ranking expression can be processed by converting into the following associated filter condition, followed by a sorting step of the answer set:

Detailed Description Text (167):
By processing the above filter condition using the strategies of the present invention described in Section 3, all of the objects with grade G or higher for a.sub.1 and v.sub.1, and for a.sub.2 and v.sub.2 can be obtained. Therefore, all of the objects with grade G or higher for the ranking expression are obtained. The number of retrieved objects in this set (i.e., two objects in the above example) should be sufficient to answer the query with the Min-based ranking expression if G has been properly estimated using the techniques described below.

Detailed Description Text (169):
where a.sub.i and v.sub.i are as before, i=1, 2, and where two objects with the top grades for the above ranking expression are desired. A grade is determined using statistics as with the processing of the Min-based ranking expression above such that there are at least two objects with that grade or higher. The ranking expression can then be processed by using the following filter expression:

Detailed Description Text (170):
By processing the above filter condition, all of the objects having grade G' or higher for the Max-based ranking expression are retrieved. If there are at least two of such objects, then the ranking expression can be answered by returning two objects with the top grades for that ranking expression from among the set of objects that are retrieved.

Detailed Description Text (171):
The key point in the mapping of the problem from a ranking problem to a (modified) filtering problem lies in finding the grade G to use in the filter condition, as in the example.

Detailed Description Text (172):
The preferred method for estimating a grade G (or G') of the translated ranking expression for the filter condition is shown in FIG. 5. In steps 74 and 76, a grade (preferably the largest) is estimated from the estimated or known fraction of objects in the repository which might satisfy the filter condition (the "selectivity") so that the number of satisfactory objects is approximately equal to or greater than the desired number of top objects requested. For any such G, the expected number of objects retrieved using the filter condition will be at least the number desired when the query is processed in step 78. Since G might be determined using selectivity estimates that might not be completely accurate, fewer than the desired number of top objects might be retrieved in step 80 by using the translated ranking expression. In this case, the test grade needs to be lowered as in step 82 and the filter condition rerun against the repository (step 78 again) until at least the desired number of top objects are retrieved and outputted in step 84.

Detailed Description Text (173):
Another way to find a grade G is to use Fagin's algorithm for processing ranking expressions as shown in FIG. 6. (See Section 4.2.) Given the number of top objects desired and input in the system in step 86, the expected number of objects that his method will retrieve for each atomic expression can be determined in step 88. These expected numbers can then be "translated" into grades in step 90 such that the expected number of objects having at least the value of the translated grade for each atomic expression will yield at least the number of expected objects. Finally in step 92, all the grades can be combined to determine a total grade G for the filter condition.

Detailed Description Text (174):
The following describes an alternate novel method of the present invention (called "Grade.sub.-- Rank") for determining grade G for the filter condition which utilizes

the number of objects desired, a ranking expression, and selectivity statistics to process the ranking expression:

Detailed Description Text (177):
3. Propagate the respective grades from the atomic expressions up to the entire ranking expression. A Min-based filter condition having a plurality of subconditions gets the minimum of the grades assigned to its subconditions. A Max-based filter condition gets the maximum of the grades assigned to its subconditions.

Detailed Description Text (180):
Consider the case where a ranking expression ranks retrieved objects according to the minimum grade of match value (which is generally between 0 and 1) between two attributes (a "Min-based ranking expression"). Suppose that the number of objects in the repository is one hundred. If only the top object is desired, the expected number of objects that Fagin's method will retrieve from searches over both attributes is ten. If it is estimated that at least ten objects will be received by choosing a grade of match for the first attribute of 0.2 and a grade of match over the second attribute of 0.3, then the Min ranking expression will yield a global grade of match G for the repository search of 0.2. This global grade value of G is then used in the filter condition derived from the translated ranking expression in Example 4.2:

Detailed Description Text (181):
Once the grades for the atomic expressions have been determined, a single global grade G can be determined and used in the filter condition for processing the ranking expression. The following example illustrates why a global grade G needs to be determined, instead of just using the atomic expression grades.

Detailed Description Text (183):
Suppose that instead of the above filter condition, the following filter conditions (for "wrong") is used for processing the ranking expression:

Detailed Description Text (184):
Assume a repository containing two objects classified by at least two attributes. The first object has a grade of match for the first attribute of 0.2 and a grade of match for the second attribute of 0.3. The second object has grades of matches of 0.25 for each of the two attributes. The second object has a Min value of 0.25 (the minimum of the two grades of match) while the first object has a Min value of only 0.2. Therefore, the second object should be the top object retrieved in the search. However, filter conditions would not retrieve the second object because its grade of match for the second subcondition is not greater than or equal to 0.3. Therefore, the above filter condition is a wrong translation of is a Min-based ranking expression.

Detailed Description Text (185):
The following material shows that processing a ranking expression by using a filter condition with grade G as determined by the Grade.sub.-- Rank method of the present invention will retrieve no more objects and thus is at least as efficient as Fagin's method.

Detailed Description Text (190):
Let R be an independent ranking expression over a continuously graded repository with O objects, and k be the number of objects desired. Let F be the filter condition associated with R that uses the grade G computed by algorithm Grade.sub.-- Rank for R and k. Then:

Detailed Description Text (215):
5.2 Ranking Expressions verses Filter Conditions

Detailed Description Text (216):
The following reports the experimental advantages of the present invention in mapping the execution of a ranking expression into the execution of a filter condition over the prior art, and particularly Fagin's method. According to Theorem 4.5, if a repository is continuously graded, there is a search-minimal execution for the filter condition that is expected to access no more objects than Fagin's algorithm for processing the ranking expression directly.

Detailed Description Text (223):
FIG. 7 shows what grade G should be used to process the filter condition associated with R.sub.Min and R.sub.Max in order to obtain at least k objects in the answer, for different values of k, where the grade granularity for computing estimates is 0.1. The

fact that selectivity statistics are not used for finer grades explains that the curves are not smooth, but have "steps." Note from the figure that R.sub.Max requires significantly higher grades than R.sub.Min. Therefore, in general, fewer objects for R.sub.Max will have to be retrieved from the conditions used for searching.

Detailed Description Text (224):
FIG. 8 shows, for each k and for R.sub.Min, the expected number of objects retrieved using the methodology of the present invention, as a fraction of the number of objects expected to be retrieved when processing the ranking expression using Fagin's methodology. FIG. 8 shows two curves, according-to the granularity of the statistics available to describe the repository, as discussed above. It is apparent from the figure that is even when only coarse statistics are available (grade granularity=0.1), the filter-condition strategy still accesses less than one third as many objects as Fagin's algorithm. This fraction gets even lower when finer statistics are available (grade granularity=0.01). One key point for explaining this behavior is that the filter-condition strategy searches objects from only one of the atomic subconditions, whereas Fagin's method retrieves objects from all of the five subconditions in the query used.

Detailed Description Text (225):
FIG. 9 shows similar results for the R.sub.Max ranking expression. In this case, the results are not as good as for R.sub.Min, because both strategies retrieve objects from all of the five atomic subconditions. Actually, when the grade granularity is 0.1, the filter-condition strategy of the present invention accesses slightly more objects than Fagin's method. This phenomenon disappears when the grade granularity is 0.01, except for k=1 and k=2. For larger values of k, the filter-condition strategy of the present invention accesses significantly fewer objects. The reason of this behavior for k=1 is that there are an expected seven objects that match expression e.sub.2 with grade one, for example. Prior art methods such as Fagin only need to retrieve one object per expression to get the top object for a Max expression. However, for k.gtoreq.3 the situation changes significantly. For example, for k=10, the filter-condition strategy accesses only an expected 30% of the objects that Fagin's method is expected to retrieve.

Detailed Description Paragraph Equation (4):
Filter Condition=(Grade(a.sub.1, v.sub.1).gtoreq.G) AND (Grade(a.sub.2, v.sub.2)(o).gtoreq.G)

Detailed Description Paragraph Equation (6):
Filter Condition=(Grade(a.sub.1,v.sub.1)(o).gtoreq.G') OR (Grade(a.sub.2,v.sub.2)(o).gtoreq.G')

Detailed Description Paragraph Equation (7):
Filter Condition=(Grade(a.sub.1,v.sub.1)(o).gtoreq.0.2) AND (Grade(a.sub.2,v.sub.2)(o).gtoreq.0.2)

Detailed Description Paragraph Table (1):

SELECT oid FROM Repository WHERE Filter condition ORDER [k] by Ranking.sub.-- expression

Current US Original Classification (1):
707/3

Current US Cross Reference Classification (2):
707/104.1

CLAIMS:

1. A method for optimizing a search over a multimedia database including a plurality of objects having at least two different searchable attributes, said method comprising:

optimizing a given filter condition, said filter condition containing a plurality of subconditions;

accessing statistics from a database based on a cost to process a particular subcondition;

choosing a subcondition to process based on the cost information;

sending a query on the chosen subcondition to retrieve all of the objects that satisfy the subcondition from the multimedia database;

evaluating the other subconditions on the retrieved objects; and

translating a ranking expression into a filter condition, and handling the processing of the ranking expression as the processing of the filter condition followed by a simple is ordering step.

2. The method of claim 1, further including requesting a specified number of top objects which fulfill the filter condition.

3. The method of claim 2 wherein the translating step includes:

estimating a grade necessary to retrieve at least the specified number of top objects requested; and

using the grade to translate the ranking expression into the new filter condition.

5. A method for optimally retrieving selected multimedia objects from a multimedia repository, the method comprising:

storing a plurality of multimedia objects in the multimedia repository;

defining at least two attributes over which the multimedia repository may be searched;

selecting a filter condition using the attributes to be searched;

searching the multimedia repository over the defined multimedia attributes;

selecting the multimedia objects which fulfill the filter condition; and

retrieving the selected objects.

10. The method of claim 5, further including:

determining a ranking expression for each object in the multimedia repository, said ranking expression representing a composite grade of match over all of the selected attributes;

requesting a desired number of objects to retrieve; and

converting the ranking expression to a filter condition; and

searching over the converted ranking expression, thus retrieving fewer useless objects than would be retrieved without using the filter condition.

13. The method of claim 10, further including

determining a first ranking expression grade from a first attribute which is expected to retrieve at least a sufficient number of objects;

determining a second ranking expression grade from a second attribute which is expected to retrieve at least a sufficient number of objects;

determining a maximum grade and associated filter condition if the ranking expression is the "Max" of the two said ranking expressions, said maximum grade being the maximum of the first ranking expression grade and the second ranking expression grade;

selecting those objects from the multimedia repository which fulfill the associated filter condition.

14. The method of claim 10, further including

determining a first ranking expression grade from a first attribute which is expected to retrieve at least a sufficient number of objects;

determining a second ranking expression grade from a second attribute which is expected

to retrieve at least a sufficient number of objects requested;

determining a minimum grade and associated filter condition if the ranking expression is the "Min" of the two said ranking expressions, said minimum grade being the minimum of the first ranking expression grade and the second ranking expression grade;

selecting those objects from the multimedia repository which fulfill the associated filter condition.

WEST

Generate Collection Print

L7: Entry 1 of 2 File: USPT Apr 29, 2003

DOCUMENT-IDENTIFIER: US 6557008 B1
TITLE: Method for managing a heterogeneous IT computer complex

Brief Summary Text (7):
As IT expands throughout the enterprise, demand for IT consultant services has concomitantly surged to address the burgeoning complexities of managing the corporate IT infrastructure. Within this growing market the complexities inherent in effectively identifying pertinent IT solutions required by a business customer and mapping those IT needs to the skills and solutions offered by a particular solution provider, the so-called Business Solutions Assessment (BSA) process, has proven to be an equally complicated endeavor, mismanagement of which may translate directly into lost business opportunities for a solution provider. As the types of IT solutions available to the business customer continually expand, IT solution providers are increasingly challenged by encounters with potential customers utilizing technologies which are outside the traditional areas supported by the provider. Alternatively, the success of a multifaceted solutions provider may be impeded by limitations in the solutions experience of a particular marketing representative in contact with a potential customer. For example, a customer who has clearly defined needs in the area of business intelligence (BI) may not be well-served by a marketing professional with experience in the area of server consolidation solutions notwithstanding the fact that his company does in fact provide BI solutions. Consequently, a successful BSA implementation would provide readily accessible support to enable a marketing professional to draw upon the entire breadth of the solution set offered by his company.

Brief Summary Text (8):
In the typical BSA scenario, the potential customer faces a number of IT challenges, some more pressing than others, in various areas of IT. It would be advantageous to enable a solutions provider representative to prioritize these needs by assessing their impact on the customer's business and to map these prioritized needs against the solutions provided by his/her company to determine which needs best matched the solutions provided thereby. At the onset of the BSA undertaking is an initial customer engagement process. During this critical period, the ability to establish credibility with the customer, and to ensure that all parties involved have a clear understanding of the business benefits which will result from the engagement and their relevance to the needs of the customer is of paramount importance. Ultimately, it is in both the customer's and provider's best business interest to qualify the project prior to investing too much resource in an endeavor. Accordingly, the ideal solution assessment process will enable the customer to make informed decision about the proffered solutions early in a project, while establishing the proper expectations, roles and responsibilities for all parties involved. Often this will entail a proof of concept undertaking to demonstrate the feasibility of the project and the creation of a high-level work plan for the project.

Brief Summary Text (10):
Other solutions have proposed the use of computer-based expert system to provide a ready interface for a consultation client. For example, U.S. Pat. No. 5,006,998 entitled Computer System With Easy Input Means For Consultation, teaches a computer system suitable for consultation in such a case where plans are to be created while a concurrent dialogue is undertaken with a client asking for advice. It distinguishes over prior art system which required that a full set of customer inputs were initially secured prior to providing a response to the customer, which suffer the disadvantage of not having facilities for addressing the customers questions which may arise in providing input to the system. The patent further teaches the inclusion of advertisements within the series of interactive inquiries addressed to the client which may additionally be responded to by the client. Accordingly, the patent is focused on the facilitation of the interaction between a computer and a client, but fails to teach

the implementation of a system or method for engagement with a client wherein increasingly detailed client data regarding a potential IT solution is interactively applied against a knowledge base to enable generation an IT business solution proposal. In other areas, U.S. Pat. No. 5,680,305 covering a System And Method For Evaluating Real Estate, teaches providing a quantitative evaluation of a real estate interest owned by a business entity via the processing of numerical data assigned to various real estate valuations such as utilization indices, and revenues of the entity owning the property to calculate a score attributable to the property, which reflects a quantitative evaluation of the real estate. This patent is a specific expert system tailored to address the intricacies of the real estate industry to place a numerical valuation on the on a given property, it does not address the complex issues involved in exploring the various facets of a customer's IT infrastructure as well as the business environment and needs goals of that customer to devise IT solutions which best meet the determined needs.

Detailed Description Text (23):
This determination process 500 then proceeds to the qualification step 505, which in a preferred embodiment is characterized by a computerized mechanism whereby a user's answers to the previously conducted survey or questionnaire are mapped against a checklist which serves to remind the provider representative to conduct certain related briefings and to elicit certain requisite information prior to proceeding further into the BSA process. For example, in qualifying a consolidation customer, the questionnaire may include queries as to the type of consolidation project to be done, i.e., hardware, application, database or other, it may also seek to ascertain whether the customer has determined a platform for the consolidation and whether the customer has selected applications for that platform. It will determine whether the customer has been briefed on the relevant technologies available from the provider and whether any outstanding technical issues remain. An illustrative list of inquiries which may be made during the qualification process 500 is provided below, however it will be understood that the qualification process serves to assist the provider's representative in deciding if a BSA is required or whether further marketing activities (see for example step 105-109 in FIG. 1) would be more appropriate at this time. It will consequently be readily appreciated that the actual approach utilized to arrive at this determination will depend to a large degree on the circumstances of the particular opportunity.

Detailed Description Text (56):
In step 904 the ranked weighted islands are then mapped against offerings from the provider so as to illustrate the type of opportunities associated with each offering. Thus, for example, the provider may notice a grouping of islands that may map to a particular offering such as consolidation whereas far fewer islands are mapped to other offerings. The figure shows an exemplary mapping in tabular form also labeled as 904.

Detailed Description Text (57):
In step 905 the resultant ranked weighted island scores having been mapped against provider offerings and are now analyzed for certain "observations" relating to the ultimate implementation of solutions for the customer. These "observations" are undertaken by the computer system and may range from the identification of actions which are required to undertake a particular opportunity, to cautions regarding potential cross-island opportunities. For example, a high score associated with number of servers may suggest a consolidation within the island, however the lack of platform-specific skills may render the consolidation within that island difficult (i.e., it would entail outsourcing or developing the skill), the tool would be implemented so as to recognize the availability of the requisite skills within another island and may provide a text generated observation pointing toward migration of the resource to the appropriate island to achieve the cost reduction goal of the customer. Many analytical implementations may be undertaken in step 705 which are considered to be within the scope of the present invention. For example, the patterns generated by the CORE tool may be compared to the results stored results of previous iterations of the BSA undertaken with other customers or with predefined models having their data stored in the database 404. This comparison may be undertaken with the aid of data mining tools such as on-line analytical processing (OLAP) tools to discern commonality among the results and previous identified opportunities.

Detailed Description Text (58):
Prior to undertaking the detailed examination 206 portion of the BSA process 200, in step 906 the CORE tool identifies the appropriate questionnaires and tools to utilize for each island in conjunction with the mapping of the weighted scores therefor. An exemplary illustration of this identification is shown as in the figure also labeled 906. This facilitates and initiates the selection 806 of one or more solutions applied

to one or more islands enabling the participants in the BSA to proceed to the final portion of the process, the detailed examination process 1000 as depicted in FIG. 10.

Detailed Description Text (82):
If it were desired to migrate a given workload from a native platform to a new platform wherein the migrated workload will be the only workload resident on the new platform (i.e., a single workload machine implementation), the newly determined maximum Tpm 1116 would be the appropriate capacity to be used to size the target machine for the migration. Accordingly, in step 1117 the maximum Tpm determined in 1116 is mapped against a look up table or otherwise mapped to determine an equivalent target machine for such a single workload machine implementation. This step 1117 is repeated for each workload to be consolidated such that if it is ultimately determined that one or more workloads is not to be migrated to a multiple workload machine, the migration solution for single workload machine implementation will be readily apparent for each workload.

Current US Original Classification (1):
707/104.1

WEST

Generate Collection Print

L2: Entry 26 of 27 File: USPT May 31, 1994

DOCUMENT-IDENTIFIER: US 5317742 A
TITLE: Dynamic translation of network management primitives to queries to a database

Drawing Description Text (9):
FIG. 8 is a flow chart of the processing of a filter item in the present invention.

Detailed Description Text (105):
The remaining processing depends on the outcome of this examination at 374. If the choice tag is a filter item, the filter item is processed at 380 and control returns to step 364. If the choice tag is AND, OR, or NOT, control passes to 382, 384 or 386 respectively where the appropriate placeholder clause is generated. Control passes from 382, 384 or 386 to step 390 where the current CMISFilter context is saved on the stack and processing begins on the next nested CMISFilter. Control then returns to step 364.

Detailed Description Text (118):
The individual processing of each filter item is described in detail in conjunction with FIG. 8 beginning at step 400. Then, the routine determines if the specified attribute ID in the filter item is multi-valued at step 402. If so, the routine processes the set-valued attribute at 404 (see FIG. 9) and exits at 408. If not, control passes to step 412 where the table and column corresponding to the particular attribute is identified. To process each individual FilterItem in the CMISFilter, the choice tag of the FilterItem is examined at step 416. If a FilterItem is a comparison to be performed with respect to a specific attribute at 416, the routine generates a SQL clause for comparison of the attribute at step 420 (see FIG. 10) and then exits at 408. If a FilterItem is an existence operation at 416, the routine generates a SQL "column-name IS NOT NULL", where column-name is the name of the column in the table corresponding to the specified attribute, and places it in the allocated buffer at 422 and then exits at 408. If a FilterItem is a substring operation at 416, the appropriate SQL clause for substring operation is generated at 426 (see FIG. 11) and the routine exits at 408.

Detailed Description Text (119):
Turning to FIG. 9, the set-valued attribute process of step 404 is described in greater detail starting at 440. The routine determines the dependent subtable in which the set-valued attribute is stored at step 442, then examines the choice tag on the filter item at 446. Process control then goes to step 450, 452 or 454 for generation of the SQL clause depending upon whether the asserted value of the choice tag is a subset of the actual value, a non-null intersection with the actual value, or a superset of the actual value respectively. The SQL clause is then placed in the buffer at 458 and the routine exits at 460.

Detailed Description Text (130):
If the choice tag is 1, a substring comparison is to be performed between the asserted value and the actual attribute value stored in the MIB. The nature of the substring comparison depends on the tag of the nested CHOICE type in the filter item:

Current US Original Classification (1):
707/3

Current US Cross Reference Classification (1):
707/4

L2: Entry 27 of 27 File: USPT May 26, 1992

DOCUMENT-IDENTIFIER: US 5117349 A
TITLE: User extensible, language sensitive database system

Detailed Description Text (25):
Line numbers 6 through 9 illustrate a "Properties" statement. The Properties statement declares the properties of symbols and are used to associate semantic tags, as set forth in the "Tags" statement (lines 13 through 15), and filters, as set forth to in the "Filter" statement (lines 17 through 20). Each semantic tag has a list of properties that it defines and each filter item has one or more lists of properties that it matches. A filter item will match all tags that define all properties that the filter item references. The Properties statement is of the form:

Detailed Description Text (29):
The argument "prefix" is used to indicate the prefix tag statement to identify the statement. The "size" argument identifies the number of tags and is used to ensure each tag statement generates the same number of tags each time the compiler is executed. The argument "property identifier" is used to match filter items to semantic tags by associating the semantic tag with properties identified in the Properties statement. For example, the Tags statement for lang1 at line 13 through 15 includes the tag "first.sub.-- on.sub.-- line" having the properties "word" and "first" and the tag "not.sub.-- first.sub.-- on.sub.-- line" having the properties "word" and "not.sub.-- first". The argument "weight" associates an integer value with each tag which is used to sort matches in response to a query. Matches having the lighter weights are presented before matches of heavier weights.

Detailed Description Text (30):
FIG. 4, lines 17-20, sets forth a Filter statement for language lang1 and lines 39-42 set forth a Filter statement for the language lang2. The "Filter" statement defines a filter panel consisting of filter items and is of the form:

Detailed Description Text (31):
When a filter item is built, the properties identified with the filter item and all semantic tags are used to figure out the semantics tags the filter item should match. The filter, which is specified for a particular source type, permits the user to narrow the search based on the semantic tags of the symbols. Preferably the filter is implemented through a menu panel displayed to the user such as the one illustrated in FIG. 5a. Optionally, the filter panel may be expanded to include additional filter items which can be used to narrow a search. For example, FIG. 5b illustrates a filter panel expanded after selection of the filter item "Declaration".

Detailed Description Text (32):
The filter panel consists of a list of filter items. Each filter item defines a list of semantic tags that the filter should match. This is built from the property list ("prop-lists") of the filter items and the semantic tags. Each filter item is either a "leaf" item or a "pull-right" item. The leaf filter item specifies one or more properties that are used to identify the semantics tags the filter item should match in order to be considered to be matches when the browsing mechanism performs a search.

Detailed Description Text (33):
Pull-right filter items specify another filter statement which is displayed when a particular filter item defining a pull-right menu item is selected and inherit the set of tags to match from the first leaf item of its pull-right filter.

Detailed Description Text (34):

The association between tags and filter items is established according to the properties of a symbol. Each semantic tag has a list of properties that it defines and each filter items has one or more list of properties that it matches. A filter item will match all semantic tags that define all properties that the filter item references.

Detailed Description Text (35):
Thus, when a user selects a filter item from the filter items provided in the browsing mechanism through the .ex file, queries will be restricted to symbols with one of the tags the filter item specifies.

Detailed Description Text (36):
The Filter statement set forth at lines 17-20 defines three filter items "All Matches", "First" and "Not First". The first filter item is defined by the property "word". The second filter item "First", is defined by the properties "word" and "first". The third filter item "Not First" is defined by the properties "word" and "not.sub.-- first". Thus, for example, if a search for is performed in a source input file written in the language lang1 and the filter item "First" is selected, the browsing mechanism will filter out all symbols except those identified by the semantic tag first.sub.-- on.sub.-- line.

Detailed Description Text (37):
The Filter statement at lines 39-42 define the filter items for the language lang2 at line 40, the filter item "All Words" is defined by the property "word". In addition, the filter item "all words" is also set to be equivalent to the filter item "All Matches" in lang1.

Detailed Description Text (39):
The statement is used to indicate that the current filter item is equivalent to the specified filter item. Thus, to display the filter, the browsing mechanism simply utilizes the filter corresponding to the filter item specified. When filter items have been equivalenced, the browsing mechanism is required to keep track of the set of matching semantic tags for each language. To do this, the .ex file, the file generated by the compiler, contains a vector filter item name, tag-vector> triples for each filter item. The browsing mechanism is then able to use the correct tag filter for the language of the source file because the database component file contains the name of the language it was generated from.

Detailed Description Text (40):
Thus, the filter item "All Words" utilizes the filter for "All Matches" in lang1. The browsing mechanism, when performing a query, for "All Matches" or "All Words" will also perform a query with respect to the other filter item.

Detailed Description Text (44):
The .ex file generated for language lang1 defined by the source file type description of FIG. 4, is illustrated by FIG. 7. At the beginning of the .ex file is a header section 700 containing a listing and location of the sections forming the file. The language section 710 identifies the languages defined. The weight section, 715, 720 specifies the weights of the semantic tags for such language. Preferably, as is the case in the present illustration, the weights are identified in the tags section 740 and 750. The filter section 730, specifies the six filter items "All Matches", "All Words", "First", "Not First", "Capitalized" and "Not Capitalized". Each filter item identifies the language the filter applies to, the name of the filter item, and the semantic tags that match the filter item. The filter items, "All Matches" and "All Words" were specified to be equivalent. This is reflected in the .ex file at section 735 where the two filter items are specified to be the same item on the filter menu. Although not illustrated, if pull-right filter items were specified in the source file type description, the filter section would reflect this by setting the "Pullright Offset" parameter of the filter item having a pullright filter to a value which identifies an offset in the file where the filter items, which are available when the pullright filter item is selected, are located.

Detailed Description Text (51):
The header section 800 contains administrative information such as the magic number of the file

(common to UNIX files) and file version numbers. The field case should be set in database component files that were generated by database builders that ignore the case of characters. The language field contains a string that is used to select the correct filter item when querying, in the present example, language lang1. The source type field indicates if the file is a top level source file or an included file. The rest of the header is a table of contents for the file where the type, size and location of all the sections are listed.

Detailed Description Paragraph Equation (6):
filter-statement:=`filter``{`<filter-item>;*`}`

Detailed Description Paragraph Equation (7):
filter-item:=leaf-item.vertline.pull-right-item

Detailed Description Paragraph Equation (8):
leaf-item:=`"`<filter-item-name>`"`[]`=`

Detailed Description Paragraph Equation (10):
pull-right-item:=`"`<filter-item-name>`"`[]`fil ter``{`*`}`

Detailed Description Paragraph Equation (11):
equiv-spec:=`equiv``"`<filter-item-name>`"`

Detailed Description Paragraph Equation (12):
equiv-spec:=`equiv``"`<filter-item-name>`"`

Current US Original Classification (1):
707/3

Current US Cross Reference Classification (1):
707/104.1

CLAIMS:

1. In a computer system comprising a CPU, input/output means and memory containing a file system, said file system comprising at least one source file comprising text, a user-extensible database system in which source files of different source file types can be input into the database system and queries can be performed on the source files input using the same browsing mechanism, said database system comprising:

source file type definition which defines the language of the source file to be input to the database system, said source file type definition comprising;

a properties statement which identifies properties of the language;

a tags statement which identifies semantic tags of the language, said semantic tags defined according to at least one property;

a filter statement which identifies at least one filter item for the language, said filter item defined according to at least one property;

compiler means for compiling the source file type definition to generate a filter description and tag set definition;

said tag set definition comprising a listing of each semantic tag for the language;

said filter description comprising a listing of the filter items for a language, each filter item comprising a filter name, the language in which the filter is used and at least one semantic tag determined according to the properties specified by the tags statement and filter statement in the source file definition;

database builder means to generate a database component file for a source file of a particular source file type, said database builder receiving as input the source file and the tag set definition for the source file type, said database builder comprising:

means for identifying each symbol in the source input file;

means for identifying the semantic tag, as set forth in the tag set definition file, for each symbol identified;

means for storing in the database component file the symbol, the line number where the symbol occurs in the source file and the semantic tag for each symbol occurrence in the source input file;

database browsing means for searching the database comprising at least one database component file in response to a query to find a symbol, said query identifying the symbol to search for and a filter item, said database browsing means comprising;

means for searching a database component file for each occurrence of the symbol identified by the query;

means for comparing the semantic tags of each occurrence of the symbol to the semantic tags specified for the filter item identified by the query;

means for generating as a result of the query those occurrences of the symbol identified by the query which has a semantic tag which matches one of the semantic tags of the filter item identified by the query;

whereby different types of source files can be input into the database system by providing a source file type definition and the same database browsing means is used to perform queries regardless of the input source file type.

5. The database system as set forth in claim 1, said filter description further comprising a menu generation means to generate a menu displaying the filter items whereby the filter items can be selected to be the filter for a query.

6. The database system as set forth in claim 1 wherein said filter description comprises a multiplicity of filter items and more than one filter item can be selected to determine the result of a query.

7. In a computer system comprising a CPU, input/output means and memory containing a file system, said file system comprising at least one source file comprising text, a user-extensible process for generating a database from source files of different source file types and performing queries on the source files input using the same browsing mechanism, said process comprising the steps of:

creating a source file type definition which defines the language of the source file to be input to the database system, comprising the steps of;

generating a properties statement which identifies properties of the language;

generating a tags statement which identifies semantic tags of the language, said semantic tags defined according to at least one property;

generating a filter statement which identifies at least one filter item for the language, said filter item defined according to at least one property;

compiling the source file type definition to generate a filter description and tag set definition;

said tag set definition comprising a listing of each semantic tag for the language;

said filter description comprising a listing of the filter items for a language, each filter item comprising a filter name, the language in which the filter is used and at least one semantic tag determined according to the properties specified by the tags statement and filter statement in the source file definition;

building a database component file for a source file of a particular source file type, comprising the steps of:

receiving as input the source file and the tag set definition for the source file type,

identifying each symbol in the source input file;

identifying the semantic tag for each symbol, said semantic tag being one of those defined in the tag set definition file;

storing in the database component file the symbol, the line number where the symbol occurs in the source file and the semantic tag for each symbol occurrence in the source input file;

searching the database comprising at least one database component file in response to a query to find a symbol, said query identifying the symbol to search for and a filter item, comprising the steps of;

searching a database component file for each occurrence of the symbol identified by the query;

comparing the semantic tag of each occurrence of the symbol to the semantic tags specified for the filter item identified by the query;

generating as a result of the query those occurrences of the symbol identified by the query which have semantic tags which match the semantic tags of the filter item identified by the query;

whereby different types of source files can be input by providing a source file type definition and queries can be performed regardless of the input source file type.

WEST

Generate Collection Print

L8: Entry 4 of 16 File: USPT Jan 22, 2002

DOCUMENT-IDENTIFIER: US 6341287 B1
TITLE: Integrated change management unit

Detailed Description Text (48):
In a similar manner, reports and other output documents exist only in the metadata created through the Java data management layer. These output documents are produced by interpreting the metadata and by extracting data from the particular business content chosen. Events may be set up based on one or more changes in the business content data, but processing of an event depends on metadata that defines the event. Processing steps can be created to summarize and "filter" data, depending upon the metadata defining the summarization and filtering techniques. Data can be imported from, and exported to, other systems based on metadata definitions of data structures.

Detailed Description Text (154):
a. Process implements processing the results of the report. This feature is used for more complicated reports, where logic or multiple data sources (Tables, Views, etc.) may be involved. A user can define one or more parameters, such as location, date or period, to filter the results of the process.

Detailed Description Text (162):
1. Run Process implements processing of results for a report and analysis tasks, such as statistical analysis of data. This function is used for more complicated reports, where logic or multiple data sources (tables, views, etc.) may be involved. A user can define one or more parameters, such as location, date or period, to filter the results of the process, can export result data, and can open reports that are based on the result data.

Detailed Description Text (181):
1. Form properties change of form level (as distinguished from field level) properties. A user can change a form layout, change the query filter condition and specify restrictions on global queries.

Detailed Description Text (410):
A "query filter condition" is a method for applying restrictions on data retrieved by a query.

Detailed Description Text (414):
A "report filter" is a method for applying restrictions on the data retrieved by a report.

Detailed Description Text (425):
The system also employs the Boolean connectives "AND" (in which both conditions must be satisfied in order for a row to be included in the result set), "OR" (in which, if either of the conditions is met, a row is included in the result set) and "GROUP" ( ),which organizes data between the parentheses for processing.

Current US Original Classification (1):
707/102

WEST

Generate Collection Print

L8: Entry 5 of 16 File: USPT Jul 17, 2001

DOCUMENT-IDENTIFIER: US 6263335 B1
TITLE: Information extraction system and method using concept-relation-concept (CRC) triples

Detailed Description Text (321):
In this further embodiment, sentences which are input in the document processing module are tagged in such a way that all of their topical content is clearly revealed. Text expresses two sorts of content: topical content and logical content. Topical content is the set of subjects which are addressed by a text, while logical content is the scaffolding which indicates how these subjects are related to one another. Topical content is conveyed by content words like "interested," "trees," "houses," "humans," etc. and phrases formed from such words, while logical content is expressed by truth-functional connectives (e.g., "and" and "not"), quantifiers (e.g., "all" and "some"), modal operators (e.g., "necessary" and "possible"), and clausal relations (e.g., "since" and "because"). Although logical content is very important for determining the truth conditions of what is said, it is not needed for defining the topics expressed by a given text. The distinction between topical and logical content can be clarified with some notions from formal logic. Topical content is the set of predicate/argument relations which characterize a text. Logical content is the set of logical operators which govern these relations.

Current US Original Classification (1):
707/5

CLAIMS:

17. The computer program product of claim 12 wherein frequency and/or recency of a CRC is used to filter or limit the number of documents reported.

WEST

Generate Collection Print

L8: Entry 7 of 16 File: USPT Mar 13, 2001

DOCUMENT-IDENTIFIER: US 6202063 B1
TITLE: Methods and apparatus for generating and using safe constraint queries

Detailed Description Text (13):
As our basic query language, we consider relational calculus, or first-order logic, FO, over the underlying models and the database schema. In what follows, L (SC, .OMEGA.) stands for the language that contains all symbols of SC and .OMEGA.; by FO (SC, .OMEGA.) we mean the class of all first-order formulae built up from the atomic SC and .OMEGA.-formulae by using Boolean connectives {character pullout},{character pullout}, {character pullout} and quantifiers .A-inverted., .E-backward. and .A-inverted.x .di-elect cons.adom, .E-backward.x .di-elect cons.adom . When .OMEGA. is (+,-,0,1,<), (+,*,0,1, <), or (+,*,e.sup.x, 0,1, <), we use the standard abbreviations FO+LIN, FO+POLY, or FO+EXP, respectively, often omitting the schema when it is understood. Regardless of whether we are in the "classical" setting, where these queries are applied to finite databases, or in the constraint query setting discussed herein, we will refer to the syntactic query languages as relational calculus with constraints .OMEGA..

Detailed Description Text (45):
In a first embodiment, the invention provides for taking an input query entered by a user and automatically translating the query into a safe query. This is accomplished in accordance with the query pre-processor 16 (FIG. 1). As will be explained and illustrated in the context of FIG. 2, the query pre-processor automatically adds restrictions or filters to the user's original query to ensure that the query is of a form that will return only finite results when provided to the spatial database engine 18. This operation is referred to herein as range-restriction. It is to be appreciated that the added restriction or restrictions may be redundant, that is, the query entered by the user may have been safe to begin with, however, by automatically adding the restrictions, the pre-processor advantageously guarantees safety.

Detailed Description Text (143):
It is to be appreciated that the pre-processor may add filters even if the query as originally entered is safe. In such case, the filters are merely redundant to the restrictions originally included by the user. In such case, the pre-processor does not affect the result by adding one or more redundant restrictions since the original query would have returned a finite result anyway.

Detailed Description Text (145):
In a second embodiment, the invention provides for taking an input query entered by a user and analyzing it with regard to safety. That is, according to the invention, the user-entered query is analyzed to determine if it has been restricted by a relation. If this is not the case, and therefore the original query is unsafe, then the user is prompted to enter one or more appropriate restrictions. Such analysis is accomplished in accordance with the query pre-processor 16 (FIG. 1). As will be explained and illustrated in the context of FIGS. 3A and 3B, the query pre-processor analyzes a user input query and, if the query is determined to be unsafe, prompts the user to add restrictions or filters to the original query to ensure that the query is of a form that will return only finite results when provided to the spatial database engine 18.

Detailed Description Text (167):
Given the above-described teachings of the invention, an illustrative scenario is presented below in the context of FIGS. 3A and 3B whereby a query pre-processor of the invention performs query analysis and then informs the user whether the original query requires any restrictions or filters to be added in order to make the query safe. We now switch to the SQL syntax again. Recall that the basic SELECT-FROM-WHERE statement corresponds to conjunctive queries, and this remains true in the presence of

constraints. Thus, safety of SELECT-FROM-WHERE SQL queries is decidable, even if one uses polynomial constraints, and has access to the hypothetical relation R that refers to all real numbers.

Current US Original Classification (1):
707/3

Current US Cross Reference Classification (1):
707/103R

Current US Cross Reference Classification (2):
707/104.1

Current US Cross Reference Classification (3):
707/2

Current US Cross Reference Classification (4):
707/4

WEST

Generate Collection Print

L8: Entry 14 of 16 File: USPT Mar 2, 1999

DOCUMENT-IDENTIFIER: US 5878431 A
** See image for Certificate of Correction **
TITLE: Method and apparatus for providing topology based enterprise management services

Detailed Description Text (275):
a host to act as a filter (optional)

Detailed Description Text (341):
The function invokes the "receive notification" method with argument RA on all topology change notification receivers registered for resource aspect RA and for all defined for resource aspect type T or for generalizations of resource aspect type T, whenever a change that corresponds to the trigger condition for that receiver is made to the topology information base. If the topology change notification receiver was defined using the optional host filter, then the function only invokes the receiver if the change occurred on the specified host.

Detailed Description Text (513):
The queries specified in this section apply to data about a resource (i.e. the aggregated view as defined through resource aspects). The resource that is being queried is identified by specifying any one of the resource aspects that represent it. The query result may be filtered so that it only contains information pertaining to a limited set of the resource aspects that represent the resource. The filter is specified by:

Detailed Description Text (516):
The output from the query will be the resource names, resource aspect types and resource aspects that the nominated resource supports, subject to the filter.

Detailed Description Text (628):
a propositional expression of resource aspect types. The propositional expression can be formed using the logical connectives AND, OR and NOT. For example, a meta-resource may specify (Type-A AND Type-B) AND NOT Type-C.

Detailed Description Text (629):
A meta-resource can optionally also express a propositional expression on resource attribute values. The propositional expression can be formed from one or more value assertions linked using the logical connectives AND, OR and NOT. For example, a meta-resource may specify (Attribute-A=25 AND Attribute-B<50) AND NOT Attribute-C="string".

Detailed Description Text (630):
The meta-resources, logical connectives and their associations form a tree. In this tree branches can only occur at the AND and OR logical connectives. Therefore meta-resource nodes have at most two associations with other meta-resource nodes:

Detailed Description Text (634):
A meta-resource can optionally specify a resource role for the associations it is involved in. If specified these resource roles limit the resources that match the meta-resource to those that take the specified role in the corresponding association. The specification for a resource role may include an index. A meta-resource can also specify a filter. A filter is a user defined method which is applied to all resources that match the meta-resource. A filter is a boolean expression that returns true or false. If a filter is specified the filter must return true for the resource to be considered as a match.

Detailed Description Text (646):

Simple navigation paths can be combined, using logical connectives, to pose more advanced queries on the topology graph structure. The logical AND, OR and NOT connectives are used to produce compound propositional expressions on topology. Evaluation of the connectives is in accordance with the rules of logic. These are:

Detailed Description Text (650):
The logical connectives extend simple and complex navigation paths at their leaf nodes by adding additional navigation paths forming more complex navigation paths. Complex navigation paths are paths within query trees. A navigation path is a path in the query tree where the root node of the navigation path is any node in the query tree.

Detailed Description Text (651):
Some examples of topology queries using the logical connectives depicted graphically in FIG. 16 may be described as follows:

Detailed Description Text (657):
All four queries start resource matching at the root meta-resource (E1). Queries Ex1 and Ex2 contains two complex navigation paths rooted at E1, whereas query Ex3 contains a single navigation path rooted at E1. Query Ex4 shows an example of nested logical connectives, with a total of three complex navigation paths that are rooted at E1.

Detailed Description Text (663):
The logical connective can be used to express compound queries on topology. A compound query combines the result of two or more queries to produce a single result. The logical AND and OR connectives are used to express compound queries. Examples of compound queries are depicted in FIG. 17 and may be described as follows:

Detailed Description Text (697):
at least one meta-resource specifying matching criteria for topology resources and optionally specifying a filter.

Detailed Description Text (710):
The API considers a navigation path to be true if there exists a topology path that matches the navigation path. The API considers that a topology path matches a navigation path that does not contain a logical NOT connective, if all of the following are true:

Detailed Description Text (714):
The API considers that a topology path matches a navigation path that contains a logical NOT connective, if all of the following are true:

Detailed Description Text (715):
all the resources in the topology path match the corresponding meta-resources in the navigation path for that portion of the navigation query that proceeds the first NOT connective in the navigation path (i.e. the first resource in the topology path matches the first meta-resource in the navigation path, and so on)

Detailed Description Text (725):
The meta-resource specifies a filter and the filter returns true when evaluated with the resource as an argument.

Detailed Description Paragraph Table (6):
media.sub.-- network 2038 this is the highest level of the low level media elements contained by a physical.sub.-- segment 2036. media.sub.-- terminator 2048 not all media technologies require termination, but if they do, here it is. media.sub.-- cable 2042 a media cable is anything which connects any two other media elements. media.sub.-- fitting 2044 this may be a barrel or tee connector in thinLAN Ethernet, a media filter in Token Ring, or anything else found between or at the end of cables. Connectors which are an integral part of the cable, like RJ-45 connectors at the end of a twisted-pair cable, are usually considered part of the cable and are not modelled separately (although there is no restriction saying they can't be). media.sub.-- connector 2046 this might be something as simple as a telephone wire punch-down block for twisted- pair, or as complex as a passive, unmanaged Token Ring MAU. media.sub.-- containment 2040 is the resource aspect type used to associate a media.sub.-- network 2038 with its containing physical.sub.-- segment 2036.

Current US Original Classification (1):
707/103R

Current US Cross Reference Classification (1):
707/1

Current US Cross Reference Classification (2):
707/101

Current US Cross Reference Classification (3):
707/3

WEST

Generate Collection Print

L8: Entry 15 of 16 File: USPT Sep 8, 1998

DOCUMENT-IDENTIFIER: US 5806061 A
TITLE: Method for cost-based optimization over multimeida repositories

Abstract Text (1):
A method for optimizing the cost of searches through a multimedia repository is disclosed where the repository contains a plurality of objects having at least two different attributes such as color in a newspaper photograph and text in the subtitle. The method comprises selecting a ranking expression, translating the ranking expression into resulting filter conditions and then optimizing the resulting filter conditions to perform the search. A database look-up step is included which determines the cost of performing searches over the various subconditions of the filter condition. The least costly subcondition is searched first to retrieve objects from the multimedia repository. The remaining subconditions are then evaluated on the retrieved objects using either a search step or probe step depending upon the determined cost to perform each. A further database look-up step predicts a grade of match necessary in the translated ranking expression to retrieve at least the number of objects requested in the search.

Brief Summary Text (4):
There are at least three major ways in which accesses to a multimedia repository differ from that to a structured database (e.g., a relational database). First, rarely does a user expect an exact match with the feature of a multimedia object (e.g., color histogram). Rather, an object does not either satisfy or fail a condition, but has instead an associated "grade of match". Thus, an atomic filter condition will not be an equality between two values (e.g., between a given color c.sub.0 and the color oid color of an object with identifier oid), but instead an inequality involving the grade of match between the two values and some target grade (e.g., Grade(color, c.sub.0 (oid)>0.7). Next, every condition on the attributes of a multimedia object may only be evaluated through an index. This is in contrast to a traditional database where, after accessing a tuple from the repository, all selection predicates can be evaluated on the tuple itself. Finally, the process of querying and browsing over a multimedia repository is likely to be interactive, and users will tend to ask for only a few best matches according to a ranking criterion.

Brief Summary Text (6):
Another example is Cypress, a picture retrieval system built using Postgres that allows a filter condition to be specified, and returns a set of objects as the answer to the filter condition. Thus, the Cypress model does not support ranking. The ranking of multimedia objects based on how well they match a ranking expression is a crucial part of our model. Each object in Cypress has an image, a set of features (e.g., color histogram), and associated text and other structured information. The querying interface supports user-defined functions and predicates including a set of predefined graded matches (e.g., a predicate "mostly yellow"). In other words, the search is somewhat boolean in that it is either true or false that certain objects match a certain predicate.

Brief Summary Text (8):
If a user of the Cypress system wishes to obtain pictures which are primarily of bodies of water, one would probably select "mostly blue" for the color attribute to obtain objects O4, O6, O7 and O9. If the user wishes to further limit the search to those pictures from Southern California locations, the user would obtain objects O4, O6 and O9. Limiting the search further to State Water Project areas would yield objects O4 and O9. Finally, the user may only wish a reservoir picture instead of an aqueduct. Typing the word "reservoir" into the text search engine of the Cypress system with the other filter conditions of the search might retrieve only object O9 from the multimedia repository containing the photographs.

Brief Summary Text (9):
The problem of optimizing user-defined filter conditions such as those in Cypress has been addressed in the prior art. Work such as that detailed in Predicate Migration: Optimizing Queries with Expensive Predicates, Proceedings of the ACM SIGMOD International Symposium on Management of Data, Washington D. C. (May 1993) focuses on conjunctive selection conditions. Techniques to optimize arbitrary boolean selection conditions have been presented in Optimizing Disjunctive Queries with Expensive Predicates, Sigmod record., 23(2):336 (1994). However, all of the above work focuses on what will be refered to herein as "probing costs", which is the cost of evaluating a condition over a given object and does not consider the search costs, which is the cost of finding out which objects satisfy a given condition. In fact, past work in this area does not consider the case where the search cost as well as the probing cost need to be considered for optimization of arbitrarily complex filter conditions containing AND/OR conjunctive and disjunctive expressions.

Brief Summary Text (10):
On the other hand, the problem of determining an optimal set of conditions to use to extract the desired set of objects (i.e., the optimal set of conditions to search) arises naturally when optimizing single-table queries with multiple indexes. The problem of sequencing the order of accesses to subfiles of transposed files is also closely related. However, in the above contexts, the probing cost is either zero or is independent of the predicates. When the filter condition is restricted to being a conjunction, the optimization problem can be cast as a join-ordering problem. However, such a formulation fails to capture characteristics that are particular of selection queries. In summary, past work in this area does not consider the case where the search cost as well as the probing cost need to be considered for optimization of arbitrarily complex filter conditions containing ANDs and ORs .

Brief Summary Text (13):
Fagin's work, however, did not include filter conditions in his model. Accordingly, a need remains for a method for searching and retrieving multimedia objects from a repository which integrate both filter conditions and ranking expressions.

Brief Summary Text (17):
The method for optimizing a search over a multimedia database, according to the invention, includes translating a user defined ranking expression into a new filter condition. Given a user specified number of multimedia objects from the data base which best fit the attributes defined by the user, one estimates a grade necessary to retrieve a sufficient number of objects to fulfill the request. These grades then define attribute subconditions in the new filter condition. Statistics are also compiled in a database which determine the costs to process various searches and probes over the attributes. This grade can be estimated over all attributes specified using a set of statistics stored about the indices. Depending upon how the condition to be used to rank the objects has been built (i.e., using the Min or the Max operators), the filter condition is processed as a conjuctive expression (AND) or a disjuctive expression (OR) to retrieve a sufficient number of objects and thereby minimizing the cost of the search.

Drawing Description Text (4):
FIG. 3 is a flow diagram showing a method for selecting a filter condition according to a designated ranking expression.

Drawing Description Text (8):
FIG. 7 is a graph showing the grade for the filter condition associated with each of the ranking expressions, as a function of the number of top objects desired.

Drawing Description Text (9):
FIG. 8 is a graph showing the expected number of objects retrieved when processing a Min-based filter condition as a function of the number of top objects desired.

Drawing Description Text (10):
FIG. 9 is a graph showing the expected number of objects retrieved when processing a Max-based filter condition as a function of the number of top objects desired.

Detailed Description Text (4):
In searching step 20, a user would define a filter condition of the search which would identify and return a candidate set (or list) of the desired type of multimedia objects. For instance, Example 2.1 below illustrates the selection of objects having

both attribute filter condition profile with a grade of 0.9 or higher and voice.sub.--sample with a grade of match of 0.5 or higher. The repository is then searched in steps 22,24 over the atomic conditions to be searched. As detailed further below in Section 3, a search minimal execution set (i.e., a set of atomic filter conditions used to retrieve multimedia objects that is both sufficient to get all the objects needed, and smallest among such sets) is determined (e.g. whether to search the repository over a particular individual or set of attribute filter conditions using "search" or to "probe" a retrieved subset of the repository). In the preferred embodiment, a statistical database is compiled using cost models for searches over different types of attributes and probe costs. Alternatively, estimation techniques can be used to estimate the expected cost to either search or probe the repository over the particular attribute. If the cost to search the repository is estimated or determined to be lower, then a search is run; otherwise, the multimedia repository is probed as in step 26. The objects which fulfill the entire filter condition are retrieved and sent back to the user's terminal (see FIG. 4) in step 28.

Detailed Description Text (5):
Turning to FIG. 3, a user practicing the invention to retrieve a desired number of suitable objects from a multimedia database in ranked order of relevance to the searching conditions can first select a ranking expression, as in step 30. As discussed further below with reference to Section 4, ranking expressions can be based on the Min or Max model (or some more complicated nested Max-Min expression) or simply by the grade of a designated attribute. In step 32, the user can designate a number which would return only a desired number of objects which best match the ranking expression (the "top objects"). In steps 34 and 36, the statistics precompiled in step 16 are accessed and a Grade is determined which is expected to be fulfilled by at least some sufficient number of top objects. A more complete explanation of this step is described in Section 4.3 below with reference to FIGS. 5 and 6. The determined ranking expression is then converted to a filter condition in step 38 according to how the ranking expression is defined (determination step 40). The repository is then searched over the filter condition (the converted ranking expression) in step 52. Finally, the returned objects are sorted based on their grade for the original ranking expression in step 54.

Detailed Description Text (7):
Conventionally, queries which specify both a filter condition (e.g., fingerprint compared to unknown print>90% match) and a ranking expression (e.g., top 20 suspects) can be executed by ranking the objects that satisfy the filter condition based on their grade of match for the ranking expression. It has been determined in simulation that the present invention, executed using the above steps, minimizes the "cost" to carry out the query; that is, it minimizes the burden on the database and search engine hardware, thereby enabling a more efficient retrieval of relevant information.

Detailed Description Text (8):
An atomic condition can be processed in two ways: by a "search", where all the objects that match the given condition are retrieved (access by value); and by a "probe", where instead of using the condition as an access method, the probe conditions are only tested against each retrieved object id (access by object id). For example, consider a filter condition consisting of a conjunction of two atomic conditions. By searching on the first condition and probing on the second, the latter benefits from the reduction in the number of objects that need probing due to the selectivity of the first condition.

Detailed Description Text (9):
The costs of these two kinds of accesses, search and probe, in multimedia repositories can vary for a single data and attribute type as well as across types. Furthermore, when the filter condition is a conjunction of atomic conditions, the problem becomes closely related to that of ordering joins. However, no work has yet studied the optimization problem when both searches and probes have non-zero costs and the filter condition is arbitrarily complex.

Detailed Description Text (10):
To optimize the processing of a filter condition when the conditions present in the filter condition are independent of one another, a space of search-minimal executions is defined whereby the search-minimal execution space is a restricted space. Nevertheless, it has been found that introducing a simple post-optimization step for conjunctive conditions yields plans that are nearly always as good as the plans obtained when plans are not restricted to be search minimal. Furthermore, experiments show that considering both the search and probe costs during query optimization impacts